Algorithms for finding transposons in gene sequences

Yue Wang* Zhongkai Zhao†

Abstract

With the process of evolution, some genes will change their relative positions in gene sequence. These "jumping genes" are called *transposons*. Through some intuitive rules, we give a criterion to determine transposons among gene sequences of different individuals of the same species. Then we turn this problem into graph theory and give algorithms for different situations with acceptable time complexities. One of these algorithms has been reported briefly as the "iteration algorithm" in Kang et al.'s paper [1] (in this paper, transposon is called "core-gene-defined genome organizational framework", cGOF). This paper provides the omitted details and discussions on general cases.

KEY WORDS: transposon, gene sequences, algorithm, graph theory

1 Introduction

A transposon is a DNA sequence that can change its relative position within the genome of a single cell. The mechanism of transposition can be either "copy and paste" or "cut and paste". However, it is not so easy to determine which genes are transposons. If there is a template sequence which can be viewed as the ancestor of other gene sequences, we can compare the other gene sequences with this template, but we cannot determine which genes change their positions since there are different ways to change the positions of genes to turn the template into the other gene sequences. In addition, when we have more than two gene sequences, we can hardly know which gene sequence is more primitive so that it can be viewed as the template.

*School of Mathematical Sciences, Peking University, Beijing 100871, P.R. China. Email address: yuewang@uw.edu

†Shandong Normal University, Jinan 250358, P.R. China.

Thus, we should describe the definition of "change its relative position" more precisely.

2 The criterion of transposons

First, let's look at an example to have an intuitive sense of transposons. Consider two gene sequences, where number represents gene's name:

{1,2,3,4} and {1,4,2,3}.

We will intuitively think gene 4 changes its position. However, changing genes $2, 3$'s positions can also explain this. Why do we think gene 4 moves, but not genes 2 and 3? The answer is that in the former situation, the number of genes changing their positions is smaller. Nevertheless, "the number of genes changing their positions" is not so easily determined. A proper way is to think of the fixed genes, the complement of transposons. The minimum of transposons corresponds to the maximum of fixed genes, so we reach **the criterion of transposons: find the longest common subsequence of these gene sequences, and the remainder are transposons.** In the example above, the longest common subsequence is $\{1, 2, 3\}$, so gene 4 is a transposon. Till now, we have turn the problem of finding transposons into finding the longest common subsequence of several number sequences. In the following sections, we will consider different situations and give corresponding algorithms.

3 Sequences without replicated genes

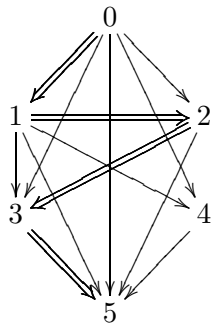
Consider gene sequences without replicated genes. For several permutations of $1, 2, 3 \cdots n$ (allowing missing some numbers), we need to find the longest common subsequence of these number sequences. Constructing a directed graph as following. Vertices are numbers from 1 to n. There exists an directed edge from vertex i to j if and only if in all sequences i appears prior to j. If there exist edges $i \to j, j \to k, \cdots, l \to m$, then the path $i \to j \to k \to \cdots \to l \to m$ corresponds to the common subsequence $\{i, j, k \cdots l, m\}$. Thus the longest path in this graph corresponds to the longest common subsequence.

This graph is transitive [2]: if there exist edges $i \to j, j \to k$, then there exists edge $i \to k$, since if i is prior to j and j is prior to$k$, then i is prior to k.

Furthermore, this directed graph has no loop. If there exists a loop $i \to j \to k \to \cdots \to l \to i$, then from transitivity there exists edge $i \to i$,

which means i is prior to i itself, a contradiction. No loop means that the longest path does exist, and its length is finite.

If we add 0 to the head of each sequence and $n+1$ to the tail, then the longest common subsequence must begin at 0 and end at $n+1$. Thus, we only need to find the longest path between two vertices in a directed graph without loop.



Graph 1

Graph 1 corresponds to the example $\{(0), 1, 2, 3, 4, (5)\}$ and $\{(0), 1, 4, 2, 3, (5)\}$, where the longest path from 0 to 5 is $0 \to 1 \to 2 \to 3 \to 5$, so the longest common sequence is $\{(0), 1, 2, 3, (5)\}$, and the transposon is 4.

We can use an iteration algorithm to find the longest path. For a vertex i, define $F(i)$ to be the vertex next to i in the longest path from i to $n+1$, and $G(i)$ to be the length of this path. Since there exists edge $i \to n+1$, F can be defined for all vertices except $n+1$. Define $G(n+1) = 0$. The longest path is $0 \to F(0) \to F^2(0) \to F^3(0) \to \cdots \to F^{k-1}(0) \to F^k(0) = n+1$, where F^j is the jth iteration of F. The iteration equations are:

$F(i) = j$, where $j \in \{k|$ there exists edge $i \to k\}$ and maximizes $G(k)$.

$$G(i) = G(F(i)) + 1.$$

Since this graph has no loop, the iteration will terminate in finite steps and produce correct answer. This iteration can be carried out though programming languages such as C++.

3.1 The space and time complexity of this algorithm

Assume there are m sequences with length n.

Space to restore these sequences is $O(mn)$. The graph is restored in an $(n+2) \times (n+2)$ matrix. Functions F and G needs $O(n)$. Since n is much larger than m in general cases, the total space complexity is $O(n^2)$.

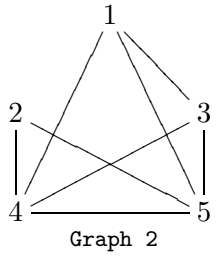
Time to construct the graph is $O(mn^2)$ since we need to check each pair $\{i, j\}$ in every sequence. In the iteration process, the calculation of each

$F(i)$ and $G(i)$ needs $O(n)$, since for every i there are at most $n+1$ edges begin at i. Thus the time complexity of iteration is $O(n^2)$. The total time complexity is $O(mn^2)$.

4 Sequences with replicated genes

In general situations there may be replicated genes in a gene sequence. Notice that the definition of transposon is a DNA sequence that has the ability to change its position, not a certain gene copy that changes its position. Thus we should regard all copies of one gene as a whole. We should not find the longest common subsequence of these sequences, like Section 3, but find the largest number of genes that all their copies remain the same relative positions.

This time we consider an undirected graph. The vertices are different genes(not all copies). There exists edge between vertices i and j (i and j are adjacent) if and only if all copies of i and j remain the same relative positions in all sequences. A group of genes with all their copies fixed corresponds 1-1 to a group of vertices that each two vertices are adjacent, which is called "complete subgraph" in graph theory. Thus we need to find the largest complete subgraph of this undirected graph. This is the famous "Maximum Clique Problem", which is NP-complete. We certainly cannot give a universal algorithm with acceptable complexity. However, transposons are much fewer compared with total genes in practice, so we can give an algorithm suitable for this situation.



Graph 2

Graph 2 corresponds to sequences $\{1,2,3,2,3,4,5\}$ and $\{2,1,3,3,2,4,5\}$, where the largest complete subgraph is $\{1,3,4,5\}$, so the longest common sequence is $\{1,3,3,4,5\}$, and the transposon is 2.

Assume there are n genes in these sequences. In graph theory, the degree of a vertex i is the number of vertices that are adjacent to i [2]. Notice that if a complete subgraph consists of k vertices, then each of these vertices must has degree not less than k. Suppose the largest degree of these vertices is k. Then we can check whether there are at least k vertices with degrees not less than k. If so, check whether there exists a complete subgraph of k vertices. If there is no complete subgraph of k vertices, replace k by $k-1$ and repeat

the former process until finding a complete subgraph. In practice, there are only a few transposons and they generally change their relative positions violently. In this situation, there will be a few vertices with significant small degrees, and all the other vertices with large degrees form the largest complete subgraph. Thus the time complexity can be sometimes limited to $O(mn^2)$.

5 More strict criterion for transposons

Maybe it is not strict enough to regard a gene which only changes its position in few sequences as a transposon. We can loosen the definition of fixed genes that they should be the longest common subsequence of $m-k$ but not all m gene sequences. Certainly, k should be small enough, such as $k < 5$. Then we can consider every $m-k$ out of m sequences and find the longest common subsequence, then pick the longest among these "longest". In practice, no matter whether there are replicated genes, the time complexity is $O(m^k n^2)$: the construction of graph is $O(mn^2)$, and finding longest common subsequence is $\binom{m}{k}O(n^2) = O(m^k n^2)$.

References

[1] Kang, Y., Gu, C., Yuan, L., Wang, Y., Zhu, Y., Li, X., Luo, Q., Xiao, J., Jiang, D., Qian, M. Khan, A.A., Chen, F., Zhang, Z. and Yu, J. (2014) Flexibility and symmetry of prokaryotic genome rearrangement reveal lineage-associated core-gene-defined genome organizational frameworks. *MBio*, **5(6)**, e01867-14.

[2] Godsil, C.D., Royle, G. and Godsil, C.D. (2001) *Algebraic Graph Theory*, Springer, New York, pp.1–4.